An Effective Local Search for Hybrid Flow Shop Scheduling Problems

Zhixiong Su[1, a], Junmin Yi[1, b]

[1]School of Management, Xiamen University of Technology, Xiamen 361024, China

[a]z.su@163.com, [b]yijunmin@xmut.edu.cn

Keywords: production scheduling; hybrid flow shop; local search; active schedule

Abstract. To solve the hybrid flow shop scheduling problems with minimum makespan objective, a local search based on the active scheduling technique was proposed. First, a good initial solution was generated by the NEH-based heuristic. Next, a problem-specific local search was developed to improve the initial solution. Last, the experimental results of benchmark instances indicate the effectiveness of the proposed algorithm, which can find the optima for more instances with a small overall average deviation of 3.445% (decreased by 2.359% compared with NEH-based heuristic).

Introduction

Production scheduling is a decision-making process that plays a critical role in manufacturing and service industries. It deals with the allocation of available production resources to tasks over given time periods, aiming at optimizing one or more objective. The hybrid flow shop (HFS) scheduling problems, as one branch of classical flow shop scheduling problems, are much more complex owing to the addition of machine assignment. This type of problems widely exists in practical production systems, such as steel, paper, electronics, petrochemical, and textile industries [1].

In this paper, we consider the HFS scheduling problems with minimum makespan (denoted as $C_{\max}$) objective. The complexity of the HFS problems has been proven to be NP-hard even when the problems have only two stages [2]. Therefore, exact algorithms such as branch-and-bound [3,4] and mixed-integer linear programming can optimally solve the small-size problems. For decades, much more effort has been devoted to searching high-quality solutions in a reasonable computational time by heuristics [5-8] and metaheuristics [9-11]. The purpose of this work is to improve the performance of an NEH-based heuristic [7], and solve the HFS problems in a way we approached using local search (LS).

Problem Formulation

The HFS scheduling problems considered in this paper can be described as follows. There are n independent and simultaneously available jobs to be processed through s stages in series. Stage k has $M^{(k)}$ identical machines ($M^{(k)} \geq 2$ for at least one stage) and has sufficient capacity of buffer storage for work-in-processes. Each machine can process at most one job at a time. Job j has a processing time p_{jk} and has to be processed without preemption by exactly one machine at stage k. Job setup times and the transportation times between consecutive stages are included in the processing times or can be negligible. The objective is to find a schedule that minimizes the makespan. The mathematical model can be found in [7,10].

Problem-Solving Strategy

Active Scheduling Technique. An active schedule is feasible schedule in which no operation can be completed earlier without delaying other operations. For the HFS problems, it is sufficient to consider only active schedules since the optimal schedule is active [12]. In order to employ this problem-specific knowledge, an extended Giffler & Thompson (EGT) algorithm was proposed to generate all possible active schedules [9].

© 2016. The authors - Published by Atlantis Press

Solution Strategy. By observing the EGT algorithm, the generation process is controlled by a set of priority rules which resolves conflict situations from stage 1 to stage s. For a given complete schedule, if all possible choices from conflict sets are considered, all active schedules close to it will be generated. This neighborhood is called active neighborhood. It is safe and efficient to limit search space to the active neighborhood.

The Proposed Algorithm

Solution Representation. In the proposed algorithm, each solution is simply represented by a string of numbers consisting of a permutation of n jobs denoted by 1, 2, …, n. This permutation-based encoding is commonly used in most of the literature for HFS [5,8,10]. Suppose that one solution is represented by (5, 4, 3, 2, 1), which means that the processing sequence is J_5, J_4, J_3, J_2 and J_1 at stage 1.

Decoding. It is notable that the solution encoding given above contains no machine assignment information in each stage. Therefore, we should consider both job sequencing and machine assignment in the decoding process. The common method used in this algorithm is as follows [11]: (1) in the first stage, schedule each job according to their sequence in the solution representation, and assign each job to the first available machine; (2) in the following stages, assign the first available machine for arriving job.

Initial Solution. The NEH was recognized as the highest performing heuristic for the permutation flow shop scheduling problems to minimize the makespan. Guinet and Solomon extended NEH to the HFS problems and outperformed the other heuristics [7]. Therefore, this NEH-based heuristic will be used to generate an initial solution.

The Framework of Local Search. The detailed steps of the proposed algorithm are as follows:

Step 1. Use NEH-based heuristic to generate a sequence $\pi_1 = (\pi_{11}, \pi_{12}, \ldots, \pi_{1n})$, and decode it into a complete schedule S.

Step 2. r_{jk} is the earliest start time at which job j can be processed at stage k, T_{mk} is the earliest available time of machine m at stage k. Set $k=1$.

Step 3. N is the set of all jobs to be scheduled at stage k, i.e. $N=\{1,2,\ldots,n\}$, and set $h=1$.

Step 4. Find the first available machine f at stage k, compute the minimum completion time $t_c^* = \min_{j\in N}\left\{\max\left(T_{fk}, r_{jk}\right) + p_{jk}\right\}$, and determine the conflict set $C=\{j \mid r_{jk} < t_c^*, j \in N\}$ for machine f.

Step 5. For $\forall j(\neq p_{kh}) \in C$, insert this job before p_{kh}, and obtain a new sequence p'_k. Schedule stage k according to p'_k and the subsequent stages (i.e. $k+1, k+2, \ldots, s$) according to the release times of the jobs. We denote the yielded complete schedule as a neighbor of S.

Step 6. Repeat Step 5 until all of the jobs in C are considered.

Step 7. Delete job p_{kh} from N, and let $h=h+1$. If $h<n$, go to Step 4.

Step 8. Let $k=k+1$. If $k \le s$, then generate the sequence p_k according to their completion times at stage k-1, and go to Step 3.

Step 9. Evaluate all of the generated neighbors and output the best one $\tilde{S}$ with a new sequence $\tilde{p}_{\tilde{k}}$ at stage $\tilde{k}$. Let $S=\tilde{S}$, $k=\tilde{k}$, $p_k=\tilde{p}_{\tilde{k}}$, and go to Step 3.

Step 10. Repeat Step 3 to Step 9 until some termination condition is satisfied.

Computational Experiments

Experimental Setup. In our computational experiments, the test instances are 77 benchmarks by Carlier and Néron [3]. The comparison was performed using two algorithms: NEH-based heuristic (simply denoted as NEH) [7], NEH with extensive search (NEHES) [8]. These three algorithms were implemented in Matlab 8.3 on a PC with Intel core i3 3.4 GHz processor and 4GB memory. In this study, the performance of the proposed LS algorithm is evaluated by two indices: (1) CPU time, (2) percentage deviation (PD) is the deviation between the solution and the lower bound (LB), i.e. $\mathrm{PD}=(C_{\max}-\mathrm{LB})/\mathrm{LB}\times 100$.

Computational Results. The numerical comparisons of NEH, NEHES and the proposed LS algorithm are given in table 1. The performances of all of the compared algorithms are summarized in table 2. For the easy instances, it can be seen from table 1 and table 2 that the LS solves to optimality 39 of the 53 instances (73.585%), with a 1.430% overall average percentage deviation. However, the NEH can solve 29 of the 53 instances (54.717%), with a larger deviation that is equal to 3.153%; the NEHES can solve 36 of the 53 instances (67.925%), with a larger deviation that is equal to 1.728%. For the hard instances, these three algorithms can solve only one of the 24 instances (4.167%), but the LS has the smallest average deviation of 7.897%. For the all 77 instances, the LS finds the optima for more instances with a small overall average deviation 3.445% (decreased by 2.359% compared with NEH). For the average computational time in which an algorithm finds the final solutions, the LS is slower than NEH and NEHES, but the average time is just equal to 0.026 s.

Table 1 Comparison results on benchmark instances (hard instances are in italic)

Instances	LB [4]	NEH			NEHES			LS		
		$C_{\max}$	PD[%]	CPU[s]	$C_{\max}$	PD[%]	CPU[s]	$C_{\max}$	PD[%]	CPU[s]
j10c5a2	88	88	0	0.011	88	0	0.009	88	0	0.011
j10c5a3	117	127	8.547	0.002	119	1.709	0.019	119	1.709	0.044
j10c5a4	121	121	0	0.001	121	0	0.001	121	0	0.001
j10c5a5	122	126	3.279	0.001	122	0	0.005	122	0	0.011
j10c5a6	110	115	4.545	0.001	112	1.818	0.006	110	0	0.022
j10c5b1	130	130	0	0.001	130	0	0.001	130	0	0.001
j10c5b2	107	107	0	0.001	107	0	0.001	107	0	0.001
j10c5b3	109	109	0	0.001	109	0	0.001	109	0	0.001
j10c5b4	122	122	0	0.001	122	0	0.001	122	0	0.001
j10c5b5	153	153	0	0.001	153	0	0.001	153	0	0.001
j10c5b6	115	129	12.174	0.001	115	0	0.003	115	0	0.002
j10c10a1	139	147	5.755	0.001	139	0	0.003	139	0	0.009
j10c10a2	158	160	1.266	0.001	160	1.266	0.014	160	1.266	0.036
j10c10a3	148	152	2.703	0.001	151	2.027	0.017	148	0	0.034
j10c10a4	149	157	5.369	0.001	149	0	0.008	149	0	0.056
j10c10a5	148	163	10.135	0.001	155	4.730	0.016	148	0	0.034
j10c10a6	146	159	8.904	0.001	151	3.425	0.017	151	3.425	0.052
j10c10b1	163	163	0	0.001	163	0	0.001	163	0	0.001
j10c10b2	157	158	0.637	0.001	158	0.637	0.015	158	0.637	0.023
j10c10b3	169	169	0	0.001	169	0	0.001	169	0	0.001
j10c10b4	159	159	0	0.001	159	0	0.001	159	0	0.001
j10c10b5	165	165	0	0.001	165	0	0.001	165	0	0.001
j10c10b6	165	165	0	0.001	165	0	0.001	165	0	0.001
j10c10c1	113	123	8.850	0.001	118	4.425	0.022	119	5.310	0.059
j10c10c2	116	127	9.483	0.001	121	4.310	0.017	121	4.310	0.069
j10c10c3	98	120	22.449	0.001	119	21.429	0.017	116	18.367	0.098
j10c10c4	103	127	23.301	0.001	127	23.301	0.015	125	21.359	0.063
j10c10c5	121	141	16.529	0.001	129	6.612	0.017	129	6.612	0.071
j10c10c6	97	113	16.495	0.001	109	12.371	0.024	106	9.278	0.109
j15c5a1	178	178	0	0.001	178	0	0.001	178	0	0.001

j15c5a2	165	167	1.212	0.001	165	0	0.004	165	0	0.003
j15c5a3	130	130	0	0.001	130	0	0.001	130	0	0.001
j15c5a4	156	156	0	0.001	156	0	0.001	156	0	0.001
j15c5a5	164	164	0	0.001	164	0	0.001	164	0	0.001
j15c5a6	178	178	0	0.001	178	0	0.001	178	0	0.001
j15c5b1	170	170	0	0.001	170	0	0.001	170	0	0.001
j15c5b2	152	152	0	0.001	152	0	0.001	152	0	0.001
j15c5b3	157	157	0	0.001	157	0	0.001	157	0	0.001
j15c5b4	147	147	0	0.001	147	0	0.002	147	0	0.001
j15c5b5	166	166	0	0.001	166	0	0.002	166	0	0.001
j15c5b6	175	175	0	0.001	175	0	0.001	175	0	0.001
j15c10a1	236	236	0	0.002	236	0	0.002	236	0	0.002
j15c10a2	200	204	2.000	0.002	204	2.000	0.031	204	2	0.059
j15c10a3	198	198	0	0.002	198	0	0.002	198	0	0.002
j15c10a4	225	225	0	0.002	225	0	0.002	225	0	0.002
j15c10a5	182	183	0.549	0.002	183	0.549	0.031	183	0.549	0.058
j15c10a6	200	201	0.500	0.002	201	0.500	0.031	201	0.500	0.055
j15c10b1	222	223	0.450	0.002	223	0.450	0.031	223	0.450	0.085
j15c10b2	187	189	1.070	0.002	187	0	0.003	187	0	0.003
j15c10b3	222	224	0.901	0.002	222	0	0.003	222	0	0.003
j15c10b4	221	221	0	0.002	221	0	0.002	221	0	0.002
j15c10b5	200	200	0	0.002	200	0	0.002	200	0	0.002
j15c10b6	219	219	0	0.002	219	0	0.002	219	0	0.002
j10c5c1	68	72	5.882	0.001	71	4.412	0.005	71	4.412	0.015
j10c5c2	74	78	5.405	0.001	77	4.054	0.006	77	4.054	0.024
j10c5c3	71	79	11.268	0.001	74	4.225	0.008	75	5.634	0.016
j10c5c4	66	74	12.121	0.001	70	6.061	0.008	70	6.061	0.023
j10c5c5	78	79	1.282	0.001	79	1.282	0.004	79	1.282	0.012
j10c5c6	69	77	11.594	0.001	71	2.899	0.008	73	5.797	0.017
j10c5d1	66	75	13.636	0.001	72	9.091	0.007	71	7.576	0.022
j10c5d2	73	79	8.219	0.001	79	8.219	0.004	76	4.110	0.037
j10c5d3	64	73	14.063	0.001	70	9.375	0.006	70	9.375	0.022
j10c5d4	70	74	5.714	0.001	72	2.857	0.006	72	2.857	0.023
j10c5d5	66	71	7.576	0.001	70	6.061	0.006	70	6.061	0.024
j10c5d6	62	69	11.290	0.001	67	8.065	0.006	66	6.452	0.021
j15c5c1	85	91	7.059	0.001	89	4.706	0.011	89	4.706	0.038
j15c5c2	90	99	10.000	0.001	94	4.444	0.017	93	3.333	0.082
j15c5c3	87	95	9.195	0.001	95	9.195	0.010	92	5.747	0.063
j15c5c4	89	98	10.112	0.001	92	3.371	0.012	92	3.371	0.056
j15c5c5	73	84	15.068	0.001	78	6.849	0.015	78	6.849	0.060
j15c5c6	91	95	4.396	0.001	95	4.396	0.009	95	4.396	0.033
j15c5d1	167	167	0	0.001	167	0	0.001	167	0	0.001
j15c5d2	82	95	15.854	0.001	91	10.976	0.012	89	8.537	0.059
j15c5d3	77	88	14.286	0.001	85	10.390	0.015	86	11.688	0.061
j15c5d4	61	90	47.541	0.001	89	45.902	0.013	88	44.262	0.060
j15c5d5	67	85	26.866	0.001	84	25.373	0.012	84	25.373	0.065
j15c5d6	79	88	11.392	0.001	86	8.861	0.014	85	7.595	0.094
Mean	-	-	5.804	0.001	-	3.800	0.008	-	3.445	0.026

Table 2 Performance summary of benchmark instances

Algorithms	Easy instances			Hard instances		
	Solved[%]	$\overline{\text{PD}}$ [%]	$\overline{\text{CPU}}$ [s]	Solved[%]	$\overline{\text{PD}}$ [%]	$\overline{\text{CPU}}$ [s]
NEH	54.717	3.153	0.001	4.167	11.659	0.001
NEHES	67.925	1.728	0.008	4.167	8.378	0.009
LS	73.585	1.430	0.021	4.167	7.897	0.039

Conclusions

A problem-specific local search was developed to solve the HFS scheduling problems. To evaluate the performance of the LS, it was tested on the well-known benchmark instances by Carlier and Néron. Computational results show that the LS outperforms the other algorithms.

Acknowledgements

This work was financially supported by the National Natural Science Foundation of China (71371162), Natural Science Foundation of Fujian Province, China (2014J01271), and High-Level Talent Foundation of Xiamen University of Technology (YSK10009R).

References

[1] R. Ruiz, J.A. Vázquez-Rodríguez, The hybrid flow shop scheduling problem, Eur. J. Oper. Res. 205 (2010) 1-18.

[2] J.N.D. Gupta, Two-stage hybrid flowshop scheduling problem, J. Oper. Res. Soc. 39 (1988) 359-364.

[3] J. Carlier, E. Néron, An exact method for solving the multi-processor flow-shop, RAIRO- Oper. Res. 34 (2000) 1-25.

[4] E. Néron, P. Baptiste, J. N. D. Gupta, Solving hybrid flow shop problem using energetic reasoning and global operations, Omega- Int. J. Manage. S. 29 (2001) 501-511.

[5] D.L. Santos, J.L. Hunsucker, D.E. Deal, Flowmult: permutation sequences for flow shops with multiple processors, J. Inf. Opt. Sci. 16 (1995) 351-366.

[6] D.L. Santos, J.L. Hunsucker, D.E. Deal, An evaluation of sequencing heuristics in flow shops with multiple processors, Comput. Ind. Eng. 30 (1996) 681-691.

[7] A. Guinet, M. Solomon, Scheduling hybrid flowshops to minimize maximum tardiness or maximum completion time, Int. J. Prod. Res. 34 (1996) 1643-1654.

[8] Z.X. Su, J.M. Yi, A two-phase heuristic algorithm for hybrid flow shop scheduling problems, J. Xiamen Univ. Tech. 23 (2015) 19-25 (In Chinese).

[9] M. Kreutz, D. Hanke, S. Gehlen, Solving extended hybrid-flow-shop problems using active schedule generation and genetic algorithms, Proc. of International Conference on Parallel Problem Solving from Nature, Springer, Paris, 2000, pp. 293-302.

[10] Z. Cui, X.S. Gu, An improved discrete artificial bee colony algorithm to minimize the makespan on hybrid flow shop problems, Neurocomputing 148 (2015) 248-259.

[11] J.Q. Li, Q.K. Pan, F.T. Wang, A hybrid variable neighborhood search for solving the hybrid flow shop scheduling problem, Appl. Soft Comput. 24 (2014) 63-77.

[12] Y.H. He, C.W. Hui, Genetic algorithm for large-size multi-stage batch plant scheduling, Chem. Eng. Sci. 62 (2007) 1504-1523.